Overview

- Agents and Environments
- Rationality
- PEAS descriptions
- Environment types
- Agent types
Reminder: Agents

- **Agent**: Entity that *perceives* and *acts*
  - Rational agents perceive and act rationally
  - Agents try to achieve goals given input perceptions (percepts)

- **Functional abstraction**: $f: \text{Percept}^* \rightarrow \text{Action}$
Agent-based design

- Regular programs are procedural
  - First, check to see how much is in the bank account
  - Debit the account by X dollars
  - Return message indicating transaction complete

- Agent-based designs are *reactive*
  - Perceive something, react a certain way
  - Similar to GUI-based programming
  - Flow comes from outside environment
    - Creates reactions within the code
Agents interact with the environment over time

Sensors/effectors do **NOT** have to be human-like

- Traffic light: sensor == treadle, effector == light
Is the camera an agent?
No, but it could be part of one
Vacuum World

- Two locations (A & B)
- Some locations have dirt
- Vacuum agent in one place at a time
Vacuum World

- Sensors: Location, Dirt?
- Effectors: Wheels, Vacuum
Vacuum World

- Percepts: InA/InB, Dirty/NotDirty
- Actions: MoveL, MoveR, Suck, NoOp
A good agent should be *autonomous*

- Actions depend on percepts
- Actions depend on *conclusions* drawn from percepts
- Actions should *never* be predetermined
  - Need to encode basic information
  - How much?
Basic Vacuum Agent Design

while (!done) {
    p = getPercept();
    a = chooseAction(p);
    doAction(a);
    done = checkIfDone();
}
Designing “chooseAction”

Could use a lookup table:

<table>
<thead>
<tr>
<th>Percept</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A,clean]</td>
<td>MoveRight</td>
</tr>
<tr>
<td>[A,dirty]</td>
<td>Suck</td>
</tr>
<tr>
<td>[B,clean]</td>
<td>MoveLeft</td>
</tr>
<tr>
<td>[B,dirty]</td>
<td>Suck</td>
</tr>
</tbody>
</table>
Evaluating the basic agent

- Does this agent perform well?
  - Need a performance measure
    - Maximize amount of dirt picked up?
    - Maximize ratio of dirt to energy expended?
    - Maximize ration of dirt to combination of energy and time?
  - Selecting a performance measure is not always easy
Evaluating the basic agent

Is this agent rational?

“For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built in knowledge the agent has.”
Evaluating the basic agent

Is this agent rational?

- “For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built in knowledge the agent has.”

- Need performance measure
- Need prior knowledge available to agent
- Need actions the agent can perform
- Need agent’s percept sequence up to date
Rationality vs. Omniscience

- We can’t expect agents to foresee everything
  - Rationality maximizes expected performance
  - If power outages are rare, we shouldn’t expect a vacuum agent to deal with it
- Agents should know what they’ve perceived and what they’ve done
  - May not know the effects of their actions
Environments

- Need to characterize the world the agent lives in
- Task environment: PEAS
  - Performance Measure
  - Environment (of the world)
  - Actuators (aka “effectors”)
  - Sensors
PEAS for Vacuum Agent

- **Performance Measure**
  - Maximize ratio of dirt to energy/time usage

- **Environment**
  - Two squares, A&B, with/without dirt

- **Actuators (effectors)**
  - Wheels, Vacuum

- **Sensors**
  - Location, Dirt Sensor
PEAS for Poker Agent

- Performance Measure
- Environment
- Actuators
- Sensors
PEAS for Survivor Agent

- Performance Measure
- Environment
- Actuators
- Sensors
Properties of task environments

- Observability
  - Can the agent get all information from the environment
  - Partially observable
    - Poker: Only the cards in your hand
    - Survivor: Votes cast at tribal council
  - Fully observable
    - Chess: Positions of all the playing pieces
    - Survivor: Who won the immunity challenge
Properties of task environments

- Determinism
  - Is the state of the world at time $t$ determined completely by the state of the world at time $(t-1)$ and the actions of the agent?
    - Deterministic?
      - Yes
    - Stochastic?
      - No, not completely predictable
    - Strategic?
      - Yes – except for the actions of other agents
      - Tic-tac-toe, Chess
Properties of task environments

- Episodic vs. Sequential
  - Episodic: only current percepts needed to decide
  - Sequential: need history and current percepts

- Static vs. Dynamic
  - Static: time halts while agent makes a decision
    - “Turn-based” games
  - Dynamic: world moves on while agent decides
    - “Real-time” applications/games
  - Semi-dynamic: world stands still, but decision time has an effect on performance score
    - Tournament-level Chess
Properties of task environments

- Discrete vs. Continuous
  - Is time handled in chunks or continuously?
  - Turns in chess vs. driving a car

- Single vs. Multi-agent
  - Does the agent have to deal with actions of others?
  - Are the competitive? Cooperative? Neither?
## Sample Environments

### Environment types

<table>
<thead>
<tr>
<th></th>
<th>Solitaire</th>
<th>Backgammon</th>
<th>Internet shopping</th>
<th>Taxi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observable??</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Deterministic?</td>
<td>Yes</td>
<td>No</td>
<td>Partly</td>
<td>No</td>
</tr>
<tr>
<td>Episodic??</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Static??</td>
<td>Yes</td>
<td>Semi</td>
<td>Semi</td>
<td>No</td>
</tr>
<tr>
<td>Discrete??</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Single-agent??</td>
<td>Yes</td>
<td>No</td>
<td>No (except auctions)</td>
<td>No</td>
</tr>
</tbody>
</table>

The environment type largely determines the agent design.

The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent.
Agent Architecture

- Five different types of agents
  - Reflex agent
  - Reflex agent with state (model based)
  - Goal-based agent
  - Utility-based agent
  - Learning agent
Condition-action rules:
if (status==dirty):
    return suck
if (location==A):
    return moveRight
if (location==B):
    return moveLeft

Lookup of action based on condition-action rules
Reflex Agent with state

Agent

Memory

Environment

State

How the world evolves

What the world is like now

Condition-action rules

What my actions do

What action I should do now

Effectors

Sensors
Reflex Agent with state

What do we mean by “state”?
- Collection of values that (partially) describes the world
- Chess: position of pieces on the board
- Automatic Traffic-light camera: color of light, visual field of camera

Agents in partially observable environments can keep track of unseen portions of the world
- Searching a maze, driving a car
- Like the “memory” of the agent
Goal-based Agent

Key difference: tries to predict effect of actions in future
Utility-based Agent

- Goal-based Agent: Succeed or not?
- Sometimes, we want something less absolute
- Utility – how “happy” am I with where I am (or where I’m going)
  - Taxi
    - Goal: arrive at destination
    - Utility: minimize time
  - Survivor
    - Goal: stay in the game
    - Utility: eliminate biggest threat
Utility-based Agent

Additional layer of “how happy will I be” on top of goal.
Learning-based Agent

- Makes Improvements
- Provides Feedback
- Suggests Exploratory Actions

Diagram:
- Critic
- Learning element
- Performance element
- Problem generator
- Sensors
- Actuators

Flow:
- Performance standard
- feedback
- changes
- knowledge
- learning goals